An Overview of the Problem of Image Segmentation with a Focus on Canny Edge Detector

Minutoli Federico, *University of Genoa, DIBRIS*

Genoa, Italy
fede97.minutoli@gmail.com

***Abstract*—Digital image processing has become a functional as well as popular research area that goes from specialized photography to several different fields such as astronomy, meteorology, computer vision, medical imaging, etc..., whose main goal is to improve the pictorial information out of an image. Contour detectors belong to different classes: (i) *Region-oriented* approaches [1], [2], in which regions of constant or slowly varying texture or color are first identified; contours are then straightforwardly determined as closed region boundaries. (ii) *Edge and line oriented* approaches, in which lines or boundaries defined by contrast in luminance, color or texture are detected. (iii) *Hybrid* approaches [3], which aim at consistency between regions and region boundaries. Here, we are interested in the approaches of class (ii). Edge detection is one of the most fundamental stages in digital image processing in many fields of applications. If the edges could be recognized accurately, all of the objects would be located efficiently and performance would be measured easily. It reduces the complexity of image processing algorithms by reducing the amount of data to be processed. An image may be affected whenever there is a jump in intensity between two adjacent pixels, and thus a crucial objective is to detect an edge while preserving the key structural properties of the image. To address this topic, Canny edge detection algorithm will be discussed in this paper w.r.t. various parameters (i.e., edge type, localization, cost, etc...). It is widely used in computer vision and medical imaging to locate sharp intensity changes and to find object boundaries in an image. It classifies a pixel as an edge if the gradient magnitude of the pixel is larger than those of pixels at both its sides in the direction of maximum intensity change. Later on, in *Section V*, a custom implementation and some exploratory analysis on sample test cases will be provided.**

***Index Terms*—Digital image processing, edge detection, feature extraction, image segmentation.**

I. Introduction

Let R represent the entire spatial region occupied by a gray-scale image. Image segmentation is the process of partitioning R into n subregions or objects, R_1, R_2, ..., R_n, such that:

(a) $\bigcup_{i=1}^{n} R_i = R$
(b) R_i is a connected set, $i = 1, 2, ..., n$
(c) $R_i \cap R_j = \oslash$ for all i and j, $i \neq j$
(d) $Q(R_i) = \top$ for $i = 1, 2, ..., n$
(e) $Q(R_i \cup R_j) = \perp$ for any adjacent regions R_i and R_j

Here, $Q(R_k)$ is a logical predicate defined over the points in set R_k and $\oslash$ is the null set. Two regions R_i and R_j are said to be *adjacent* if their union forms a connected set. Thus, we see that the fundamental problem in segmentation is to partition an image into regions that satisfy the preceding conditions. Segmentation algorithms for monochrome images generally are based on one of two basic categories dealing with properties of intensity values: *Discontinuity* and *similarity*. In the first category, the assumption is that boundaries of regions are sufficiently different from each other and from the background to allow boundary detection based on abrupt local discontinuities in intensity. Edge-based segmentation is the principal approach used in this category. Region-based segmentation approaches, in the second category are based on partitioning an image into regions that are similar according to a set of predefined criteria. Thresholding, region growing, and region splitting and merging are examples of methods falling in this category. Focusing on the detection of local changes in intensity, the three types of image features we are interested in are isolated points, lines and edges. *Edge pixels* are pixels at which the intensity of an image function changes abruptly and *edges* are sets of **connected** edge pixels. Edge detectors are thus local image processing methods designed to detect such defined edge pixels. Lines and isolated points may be viewed as thin edge segments, in which the intensity of the background on either side of the line is either much higher or much lower than the intensity of the line pixels.

II. State of the Art

Edge detection has been one of the most explored tasks in image processing for the last two decades. Nonetheless, no known algorithm has been able to cope efficiency and reliability so far, due to various practical reasons, such as possible low signal-to-noise ratio (SNR) or presence of textures in an input image. It follows the necessity to interpret the input image by introducing prior high-level knowledge, such as shape, about the contours to be detected. However, such integration of low- and high-level vision cues turns out to be a difficult task, which usually leads to very complex models and computationally demanding algorithms, that sometimes resort on machine learning approaches, like CNNs. Since human judgment is the only possible criterion that can be used in order to say if a given visual feature is a contour or not, contours are operationally defined as the set of lines that human observers would consent on to be as such in a given image. It follows that research in contour detection aims at understanding and modeling mathematically the features which people (consciously or unconsciously) use to recognize such line sets. Specifically, contour detection algorithms are

classified according to the perceptually important features that are used in each approach. Most of them belong to the *Local methods* class, in which the defining features are local discontinuities in luminance, color and texture among neighbor pixels. Such discontinuities can often influence the human visual perception of the whole image and not only the immediate neighborhood of the boundary, as it is shown in Fig. 1. Here, indeed, the two object icons are identical in 95% of their pixels but the tiny differences in their contours lead to the perception of two objects that are semantically completely different. To cope with this issue, *Contextual and global methods* have been proposed, in which contours are identified based on good continuation and closure, such as edge grouping according to Gestalt laws or active contours.

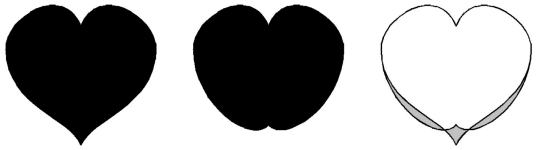


Fig. 1: Semantically different images

Local methods' class is really vast, yet three main subclasses can be identified: (i) *Differential*, which focuses on basic concepts, but still offers great speed suitable for real-time applications (i.e., Zero-crossings, Marr-Hildreth algorithm [4], Canny optimal edge detector [5], etc...); it will be the focus of the discussion in Section III. (ii) *Statistical*, which focuses on the extraction of texture boundaries (i.e., Distribution of gray-levels [6], distribution of color attributes [7], etc...) (iii) *Phase congruency*, which focuses on idempotent regions of a given image (i.e., Quadratic filters [8], [9], etc...)

Preprocessing aimed at reducing textural information and noise in a given image, as well as quantitative performance evaluation, are also issues that have not received sufficient attention. A general framework called *Adaptive smoothing* [24] consists of computing a local weighted average of the gray levels on a neighborhood of each pixel P, where the weights can depend in several ways on the local pattern configuration, including (i) *Bilateral filtering* [10], (ii) *Mean shift*, (iii) *Vector order statistics* (VOS) [11]. VOS filters are of particular interest because they help reducing the effect of impulsive and Salt-and-pepper noise. More on this in Section IV, when I'll talk about recent explorations of Canny.

III. Differential methods

Local averaging at pixel level smooths an image. Given that averaging is an operation analogous to integration, it should come as no surprise that abrupt changes in intensity can be detected using derivatives. For reasons that will become evident shortly, first- and second-order derivatives are particularly well suited for this purpose.

Derivatives of a digital function are defined in terms of differences and they will be explored here for simplicity. There are various ways to approximate these differences but, as explained in [12] any approximation used for a first derivative has three requirements:

- Must be zero in areas of constant intensity
- Must be nonzero at the onset of an intensity step or ramp
- Must be nonzero at points along an intensity

An approximation to the first-order derivative is obtained at point x of a one-dimensional function $f(x)$ by expanding the function $f(x + \Delta x)$ into a Taylor series over x, letting $\Delta x = 1$, and keeping only the linear terms. This results into:

$$\frac{\partial f}{\partial x} = f'(x) = f(x+1) - f(x) \tag{1}$$

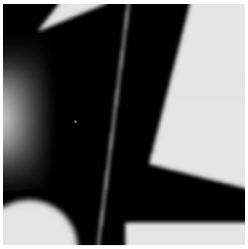
Similarly, an approximation used for a second derivative:

- Must be zero in areas of constant intensity
- Must be nonzero at the onset and end of an intensity ramp
- Must be zero along intensity ramps

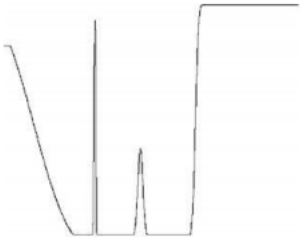
Images can be seen as digital quantities whose values are finite, hence the maximum possible intensity change is also finite, and the shortest distance over which a change can occur is between adjacent pixels. This results into:

$$\frac{\partial^2 f}{\partial x^2} = f''(x) = f(x+1) + f(x-1) - 2f(x) \tag{2}$$

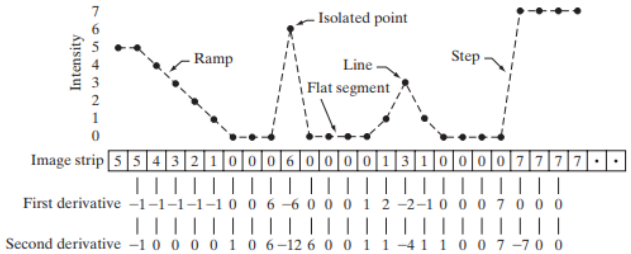
Fig. 2a shows an image that contains various solid objects, a line, and a single noise point. It follows in Fig. 2b a horizontal intensity profile of the image approximately through its center, including the isolated point. It ends in Fig. 2c with a simplification of the intensity levels profile over 3 bits, with just enough points to make it possible to analyze numerically how the first- and second-order derivatives behave as they encounter a noise point, a line, and the edges of objects.



(a) Original image



(b) Intensity levels profile

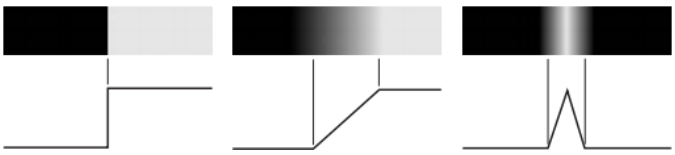


(c) Quantized statistics

Edges are classified in three types:

- *Step edges* - Transition between two intensity levels occurring ideally over the distance of 1 pixel.

- *Ramp edges* - Slow and graduate transition between two intensity levels, typically seen in practice due to the presence of blurring and noise in the images.
- *Roof edges* - Transition to a different intensity level and back to itself, typical of range imaging.



(a) Step edge (b) Ramp edge (c) Roof edge

Conclusions can be made by comparing Fig. 2 and Fig. 3, on how the first- and second-order statistics deal with capturing the edges: 1) First-order derivatives generally produce thicker edges in an image. 2) Second-order derivatives have a stronger response to fine detail, such as thin lines, isolated points, and noise. 3) Second-order derivatives produce a double-edge response at ramp and step transitions in intensity. 4) Zero-crossings of the second derivative can be used to determine whether a transition into an edge is from light to dark or dark to light since it follows its sign.

Spacial filters are the approach of choice for computing first and second derivatives at every pixel location in an image. For the 3×3 filter mask in Fig. 4, the procedure is to compute the weighted sum of products of the mask coefficients with the intensity values in the region encompassed by the mask w.r.t. the center point of the region:

$$R = w_1 z_1 + w_2 z_2 + ... + w_9 z_9 = \sum_{k=1}^{9} w_k z_k \qquad (3)$$

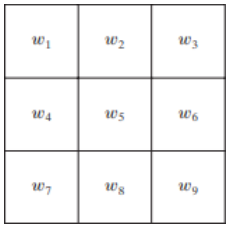


Fig. 4: Spacial filter mask

Both orders of derivatives thus have their use in extracting the edges out of an image. Which one to choose is the question, though, and it often relies on how sensitive to noise the image is. Little to none visual noise in the image can still have a significant impact on them as Fig. 5 clarifies.

Basic edge detection will be covered next, highlighting the main approaches, tools and masks a generic edge detection algorithm follows. This results in a 3-steps process that will be analyzed with greater detail in Canny's algorithm:

- Image smoothing for noise reduction
- Detection of edge points
- Edge localization

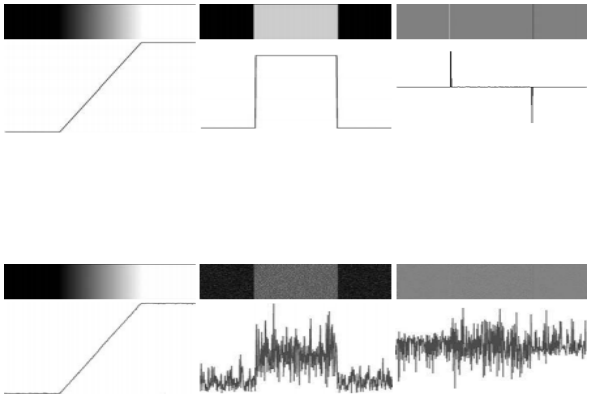


Fig. 5: (i) Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0 and 1.0 intensity levels, respectively. (ii) First-derivative and intensity profiles. (iii) Second-derivative and intensity profiles

A. Basic edge detector

As illustrated previously, detecting changes in intensity for the purpose of finding edges can be accomplished using first- or second-order derivatives. The tool of choice for finding edge strength *and* direction at location (x, y) of an image function, f, is the gradient, denoted by ∇f and defined as:

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix}$$

∇f has the important geometrical property that it points in the direction of the greatest rate of change of f at location (x, y). Its *magnitude* denoted as $M(x, y)$, where

$$M(x, y) = |\nabla f| = \sqrt{g_x^2 + g_y^2}$$

is the value of the rate of change along the *direction* of the gradient vector, which is given by the angle:

$$\alpha(x, y) = \arctan \begin{bmatrix} g_y \\ g_x \end{bmatrix}$$

As in the case of the gradient image, $\alpha(x, y)$ also is an image of the same size as the original one created by the array division of g_x by g_y. The direction of an edge at an arbitrary point (x, y) is *orthogonal* to the direction, $\alpha(x, y)$, of the gradient vector at the point, also referred to as the *edge normal*.

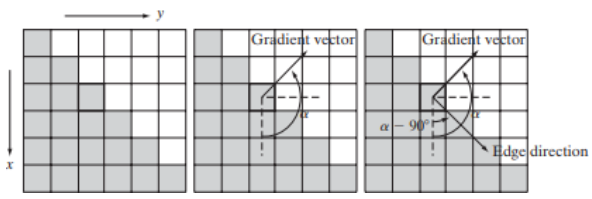


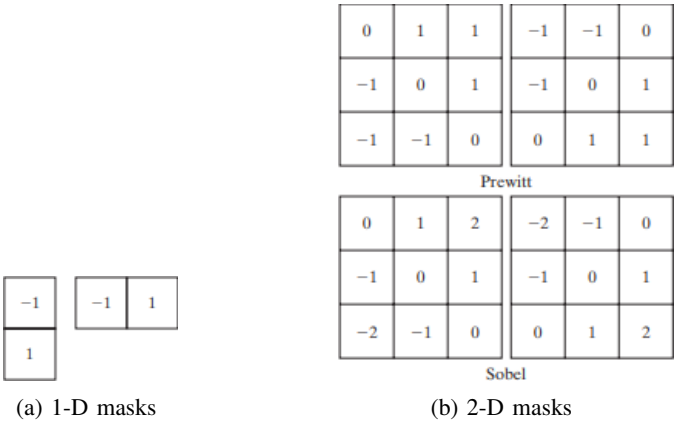
Fig. 6: Edge strength and direction at (x, y).

Since f is a discrete function, we apply (1):

$$g_x = f(x+1, y) - f(x, y) \tag{4}$$
$$g_y = f(x, y+1) - f(x, y) \tag{5}$$

which can be implemented with the 1-D masks in Fig. 7a

Diagonal edge direction is trickier instead, and requires the use of a 2-D mask. Masks of size 2×2 are simple conceptually, but they are not as useful for computing edge direction as masks that are symmetric about the center point, the smallest of which are of 3×3 size. These masks take into account the nature of the data on opposite sides of the point and thus carry more information regarding the direction of an edge.



(a) 1-D masks (b) 2-D masks

Fig. 7: Gradient operators

Fig. 7b shows two of the most prominent 3×3 operators in the literature, *Prewitt* and *Sobel* masks, that have also been compared in the Jupyter notebook. Even though the former is a little simpler to implement, the latter is preferable due to having better noise suppression.

These operators work best when an averaging filter has been previously applied to the image function f. The bigger the filter the less sensitive to fine detail the operators will be, letting them focus on the principle edges. Hence, sometimes a 5×5 averaging filter is preferred to a 3×3 one. Another approach aimed at achieving the same basic objective is to threshold the gradient image:

$$g(x, y) = \begin{cases} 1, & \text{if } M(x, y) >= T \\ 0, & \text{otherwise} \end{cases}$$

where T is often set equal to 33% of the maximum value of the gradient image. This results in pixels with values greater than T shown in white, while those below it in black. More complex approaches, such as *double thresholding*, are of common use and I will later introduce Otsu's method at finding the high threshold T_h which I used in my implementation of Canny.

Moreover, the implementation of the gradient I just presented is not always desirable because of the computational burden required by squares and square roots. An approach used frequently is to approximate it with absolute values:

$$M(x, y) \approx |g_x| + |g_y|$$

This is more attractive computationally, and it still preserves relative changes in intensity levels. The price paid for this advantage is that the resulting filters will not be *isotropic* (invariant to rotation), in general.

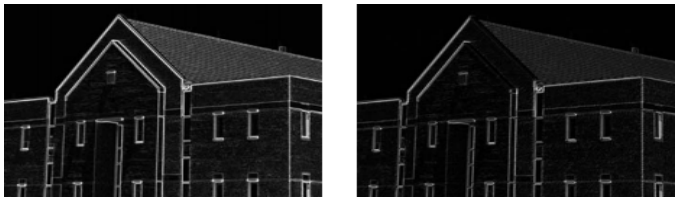


Fig. 8: (i) Edges map pre-thresholding. (ii) Edges map after-thresholding. As it is possible to see, even if some small parts of the pediment are lost after applying thesholding at T, all the texture on the roof is gone and the detector can focus solely on the primary edges over the facade. This applies for images with shadows, as well, since they need to be discarded.

It's worth mentioning an approach a little more advanced than what I just presented, but yet precedent to Canny, known as the Marr-Hildreth edge detector. Its main difference is that it finds edges on the zero-crossings of a second-order term, called *Laplacian of Gaussian* (LoG), with a discernable *Mexican hat* shape. They noted that it is possible to approximate the LoG filter in by a *difference of 2-D Gaussians* (DoG):

$$DoG(x, y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}$$

IV. Canny algorithm

A. Mathematical derivation

Canny edge detector represents a really good approximation of the optimal edge detector, that is, and edge detector which satisfies the three criteria Canny himself declared as relevant towards the performance. Only two criteria were set initially:

1) *Good detection* - There should be a low probability of failing to mark real edge points, and low probability of falsely marking non-edge points. Since both these probabilities are monotonically decreasing functions of the signal-to-noise ratio (*SNR*), this criterion corresponds to maximizing the SNR itself.
2) *Good localization* - The points marked as edge points by the operator should be as close as possible to the center of the actual true edge.

This criteria have been formulated in [5] in a mathematical closed-form which can be used to design detectors for arbitrary edges, with the appropriate replacements. It was empirically discovered, though, that the first two criteria were not "tight" enough, and thus it was necessary to add a third one:

3) *Single response to edges* - This should have been implicitly captured by the first criterion since when there

are two responses to the same edge, one of them must be considered false. However, this did not happen and it had to be made explicit.

To facilitate the analysis, 1-D edge profiles are considered. This assumes two dimensional edges to locally have a constant cross-section in some direction. Images are defined as the edge function $G(x)$ plus additive white Gaussian noise.
The problem is formulated as follows: (i) It's given an edge of known cross-section bathed in white Gaussian noise as in Fig. 9a. (ii) It is then convoluted with a filter whose impulse response, $f(x)$, could be illustrated by Fig. 9b, with the output in Fig. 9c given by the formula $H_G = \int_{-W}^{+W} G(-x)f(x)dx$. (iii) Its center is marked at a local maximum in the output of the convolution. Which filter $f(x)$ gives the best performance w.r.t. the above criteria is the question.

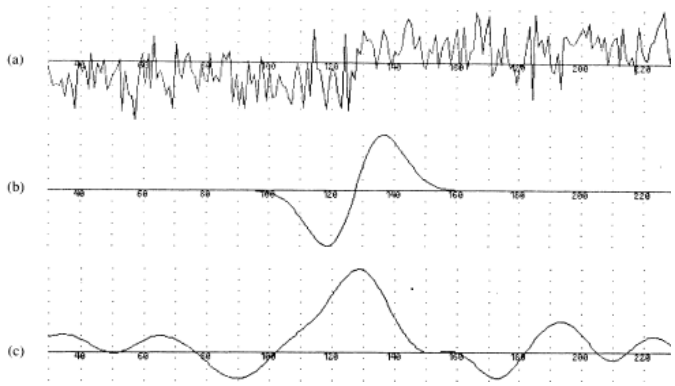


Fig. 9: Canny problem design

Given the filter response to the noise $n(x)$, as $H_n = n_o\sqrt{\int_{-W}^{+W} f^2(x)dx}$ with n_0^2 the mean-squared noise amplitude per unit length, the first criterion is simply given by:

$$SNR = \frac{\left|\int_{-W}^{+W} G(-x)f(x)dx\right|}{n_o\sqrt{\int_{-W}^{+W} f^2(x)dx}} \tag{6}$$

The reciprocal of the root mean squared (*RMS*) distance of the marked edge from the center of the true edge is used instead as a measure for the second criterion, as it increases as it improves. Since the edges are marked at local maxima in the response of the operator $f(x)$, the first derivative of the response will be zero at these points. Thanks to one Taylor expansion and the symmetry of the involved response functions $H_G(x)$ and $H_n(x)$, the RMS distance at a given local maximum x_0 is defined as the reciprocal of an approximation to the standard deviation of x_0, known as δx_0, w.r.t. the first derivative $f'(x)$. Thus, it becomes:

$$Localization = \frac{\left|\int_{-W}^{+W} G'(-x)f'(x)dx\right|}{n_o\sqrt{\int_{-W}^{+W} f'^2(x)dx}} \tag{7}$$

Equations (6) and (7) are mathematical forms for the first two criteria, and the design problem reduces to the maximization of both of these simultaneously. In order to do so, the product of (6) and (7) needs to be maximized.

$$\Theta(f) = SNR \times Localization(f) \tag{8}$$

The SNR criterion does not take into account the behavior of the filter nearby the edge center, though. From the Schwarz inequality for integrals we can show that (6) is bounded above by $n_0^{-1}\sqrt{\int_{-W}^{+W} G^2(x)dx}$, and (7) follows with $G'^2(x)$. Both bounds are attained, and the product of (6) and (7) is maximized when $f(x) = G(-x)$ in $[-W, W]$. Thus, according to the first two criteria, the optimal detector for step edges is a truncated step, or difference of boxes operator. However, looking at these differences closely, it has a very high bandwidth and tends to exhibit many maxima in its response to noisy step edges, which is a serious problem when the imaging system adds noise or when the image itself contains textures. Its response to a noisy step is roughly a triangular peak with numerous sharp maxima in the vicinity of the edge that are so close together that it is not possible to identify one as the response and the others as noise. It would be ideal if the distance between peaks in the noise response approximated the width W of the response of the operator to a single step. In order to express this as a functional constraint on $f(x)$, an expression for the distance between adjacent noise peaks is needed. Since the mean distance between adjacent maxima in the output is twice the distance between adjacent zero-crossings in the derivative of the operator, the auto-correlation function $R(\tau)$ can be used to look for zero-crossing for the function $f'(x)$. The distance between adjacent maxima in the noise response of $f(x)$ is then set to a fraction k of the operator width W. Eventually, the expected number of noise maxima in a region is given by the quantity $N_n = \frac{2}{k}$. It is important to note that even though the inter-maximum spacing between zero-crossings in $f'(x)$ scales with the operator width W, it can be proved that for any fixed k, the third criterion is invariant w.r.t. spatial scaling of f.

In general it will be difficult to impossible to find a closed-form for the function f which maximizes (8) subject to this last constraint. Even when G has a particularly simple form, the form of f may be complicated. However, if we are given a candidate function f, evaluation of (8) and of N_n is straightforward. In particular, if the function f is represented by a discrete time sequence, evaluation of (8) requires only the computation of four inner products.

Penalty methods can be implemented, as well, as additional constraints, such that for each constraint a penalty function is defined which has non-zero value $P_i(f)$ when one of the constraints is violated. Equation (8) can then be rewritten as:

$$\Theta(f) = SNR \times Localization(f) - \sum_i \mu_i P_i(f) \quad (9)$$

Canny's original paper [5] then focuses to the case where the input $G(x)$ is a step edge. Specifically we set $G(x) = Au(x)$ where $u(x)$ is the *Heaviside* step function and A is the amplitude of the step. By substituting $G(x)$ in (6) and (7):

$$SNR = \frac{A}{n_0}\Sigma(f), \quad \Sigma(f) = \frac{\left|\int_{-W}^{0} f(x)dx\right|}{n_o\sqrt{\int_{-W}^{+W} f^2(x)dx}} \quad (10)$$

$$Localization = \frac{A}{n_0}\Delta(f), \quad \Delta(f) = \frac{|f'(0)|}{n_o\sqrt{\int_{-W}^{+W} f'^2(x)dx}} \quad (11)$$

where the terms $\Sigma(f)$ and $\Delta(f)$ can be seen as two performance measures which depend on the filter only.

Supposing now that we form a spatially scaled filter f_W from f with width W, we can substitute f_W in (10) and (11) to obtain the performance of the scaled filter:

$$\Sigma(f_W) = \sqrt{W}\Sigma(f) \quad \text{and} \quad \Delta(f_W) = \frac{1}{\sqrt{W}}\Delta(f') \quad (12)$$

This tells us two things: (i) A filter with a broad impulse response will have better SNR than a narrow filter, which I am going to verify in *Section V* when I'll compare Gaussian filters of 3×3 and 5×5 size. (ii) Localization is better with a narrow filter than a broad one. What is surprising is that the changes are inversely related, that is, both criteria either increase or decrease by $\sqrt{W}$. Thus, there is an uncertainty principle relating the detection and localization performance of the step edge detector. Through spatial scaling of f over a window W we can trade-off detection performance against localization, but we cannot improve both simultaneously. Then the natural choice for the optimal composite criterion, would be the product of (10) and (11), since it would be invariant under changes in scale:

$$\Sigma(f)\Delta(f') = \frac{\left|\int_{-W}^{0} f(x)dx\right| |f'(0)|}{\sqrt{\int_{-W}^{+W} f^2(x)dx \int_{-W}^{+W} f'^2(x)dx}} \quad (13)$$

The solutions to the maximization of this expression will be a *class of functions* all related by spatial scaling.

Further derivation is going to include approaching this problem with Lagrange multipliers under stricter conditions, which leaves some λ_i parameters to be computed from the following expression:

$$\Psi(x, ff', f'') = f^2 + \lambda_1 f'^2 + \lambda_2 f''^2 + \lambda_3 f \quad (14)$$

Canny also discovered that there will be only one maximum near the center of an edge if $|Af'(0)|$ is greater than the second derivative of the response to noise only. This latter quantity is a Gaussian random variable which enables us to form a rough estimate for the probability of a spurious maximum in the neighborhood of the true maximum, as follows:

$$P_m = 1 - \Phi\left(\frac{A}{n_0}\Sigma\right) \quad (15)$$

given in terms of the normal distribution function Φ.

The following discussion is maybe the most valuable, out of all the things presented in Canny's paper. Equation (13) can be approximated by the first derivative of a Gaussian $G'(x)$, where $G(x) = e^{\left(-\frac{x^2}{2\sigma^2}\right)}$. One of the main reasons for doing this is that there are very efficient ways to compute the 2-D extension of the filter if it can be represented as some derivative of a Gaussian distribution. Thus, it is of great interest to compare the optimal operator in (13) with the theoretical performance of the first derivative of a Gaussian filter with impulse response:

$$f(x) = -\frac{x}{\chi^2}e^{\left(-\frac{x^2}{2\sigma^2}\right)} \quad (16)$$

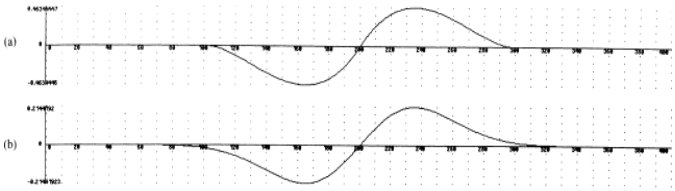


Fig. 10: (a) Optimal step edge detector.
(b) First derivative of a Gaussian.

The performance of the first derivative of Gaussian operator above is worse than the optimal operator by about 20% and its multiple response measure R, is worse by about 10%. It would probably be difficult, though, to detect a difference of this magnitude by looking at the performance of the two operators in Fig. 10, and because the first derivative of the Gaussian operator can be computed with much less effort in 2-D, it has been used widely in applications.

The amplitude distribution of the filter response can also be used in order to gain better separation between signal and noise since it tends to be different for edges and noise. By Canny's model, the noise response should have a Gaussian distribution, while the step edge response will be composed of large values occurring very infrequently. *Streaking*, though, can occur even when noise estimation techniques are in use, if only a single threshold T is used. The edge detector, basically, will be susceptible to breaks up of the edge contour caused by the operator output fluctuating above and below T along the

length of the contour. Double tresholding patches this issue up, and thus it's typically the preferred choice. Streaking's probability is here greatly reduced because for a contour to be broken it must now fluctuate above the high threshold and below the low threshold.

In 1-D the position of a step edge can be characterized in space with one position coordinate. In 2-D an edge also has an orientation, which Canny defines as the tangent to the contour that the edge highlights in 2-D. To detect edges along a particular orientation Canny suggests to create a 2-D mask for this orientation by convolving a linear edge detection function aligned normal to the edge direction with a projection function parallel to the edge direction. Substantial savings in computational effort are possible if the projection function is a Gaussian with the same σ as the first derivative of the Gaussian used as the detection function. It is possible to create such masks by convolving the image with a symmetric 2-D Gaussian and then differentiating normal to the edge direction. This is not needed in every direction, also, because the slope of a smooth surface in any direction can be determined exactly from its slope in two directions.
Suppose we wish to convolve the image I with an operator G_n which is the first derivative of a 2-D Gaussian G along some direction $\mathbf{n}$, i.e.,

$$G = e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} \qquad G_n = \frac{\partial G}{\partial \mathbf{n}} = \mathbf{n} \cdot \nabla G \tag{17}$$

Ideally, $\mathbf{n}$ should be oriented normal to the direction of an edge to be detected, and although this direction is not known *a priori*, a good estimate of it can be computed from the smoothed gradient direction:

$$\mathbf{n} = \frac{\nabla(G \circledast I)}{|\nabla(G \circledast I)|} \tag{18}$$

An edge point is defined to be a local maximum along the direction $\mathbf{n}$ of the operator applied to the image I. At a local maximum, we have a 0:

$$\frac{\partial}{\partial \mathbf{n}} G_n \circledast I = 0 \tag{19}$$

and substituting for G_n from (17) and associating Gaussian convolution, the above becomes:

$$\frac{\partial^2}{\partial \mathbf{n}^2} G \circledast I = 0 \tag{20}$$

The edge strength, at such an edge point, will be:

$$|G_n \circledast I| = |\nabla(G \circledast I)| \tag{21}$$

This formulas to compute the edge magnitude and direction will be prove in the *Implementation* subsection to be invaluable building blocks to design the Canny algorithm.

B. Recent exploration

Canny edge detector's main flaw is that it applies the same Gaussian smoothing effect both to noise and true edges, augmenting the risk of detecting wrong edges (known as *False positives*) or missing true edges (known as *False negatives*). Many approaches have been proposed to mitigate such issue, and I am going to touch upon two of those:

- Bing adaptive filter [13] evaluates the discontinuity between gray-scale values among adjacent pixels following a sort of *Fuzzy logic* paradigm. Higher discontinuities are assumed near the edges, thus a weight close to 0 is assigned to the filter. Low discontinuities, instead, are assumed when in presence of noise, thus a weight close to 1 is assigned to the filter to achieve a more accurate smoothing effect.
- A subsequent improvement to Canny using the above adaptive approach has been proposed [14] by applying an adaptive Gaussian filter to smooth the noise and edges out differently and employing variable σ and threshold values T on different parts of the image, referred from here on as sub-images. Mean value across a given sub-image is computed and depending upon these values each sub-image will be processed with Bing's adaptive filter tuned on different σ and threshold values T. Indeed sub-images having very high or very low mean values suggest the presence of weak edges, while those with intermediate mean values are likely to have prominent edges.

C. Implementation

I am going to present below in Fig. 11 a block schema of my implementation of Canny edge detector. Please refer to the attached Jupyter notebook for the details on each of those steps. Indeed, I have briefly analyzed for all of them the usefulness, the mathematical concepts and few possible alternatives towards the goal of an optimal (w.r.t. Canny constraints) edge detector.

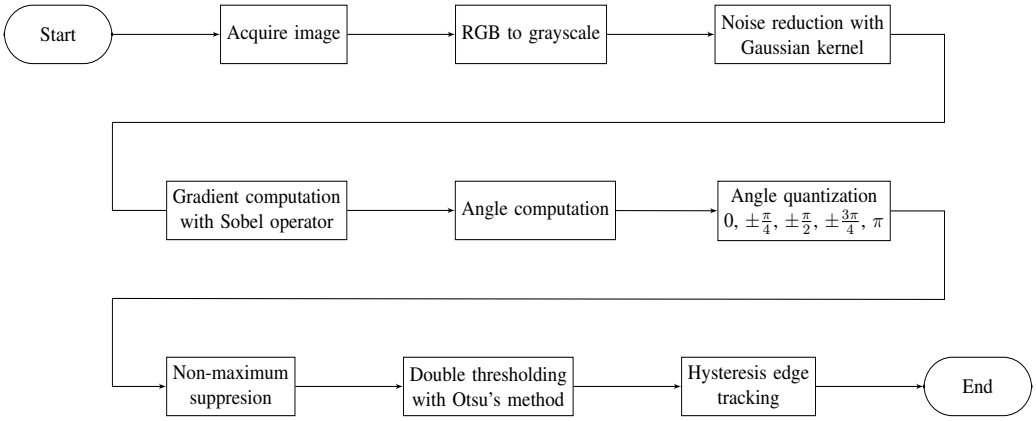


Fig. 11: Canny edge detector workflow

V. Experiments and results

In this section I am going to present the most interesting exploratory results I got out of the many tests I had on *images/lena.png* and on *images/butterfly.png* in the Jupyter notebook. Given the high light contrast, the few shades and the bright colors they have (Fig. 12a), I thought they were really good images to show the capabilities of Canny. As I said earlier noise reduction is a key step towards the goal of Canny edge detection, and in general; thus, I tried both with a 3×3 and a 5×5 Gaussian filter, to verify that the bigger the filter is, the higher the smoothing effect.

(a) Grayscale image

(b) 3×3 Gaussian filter

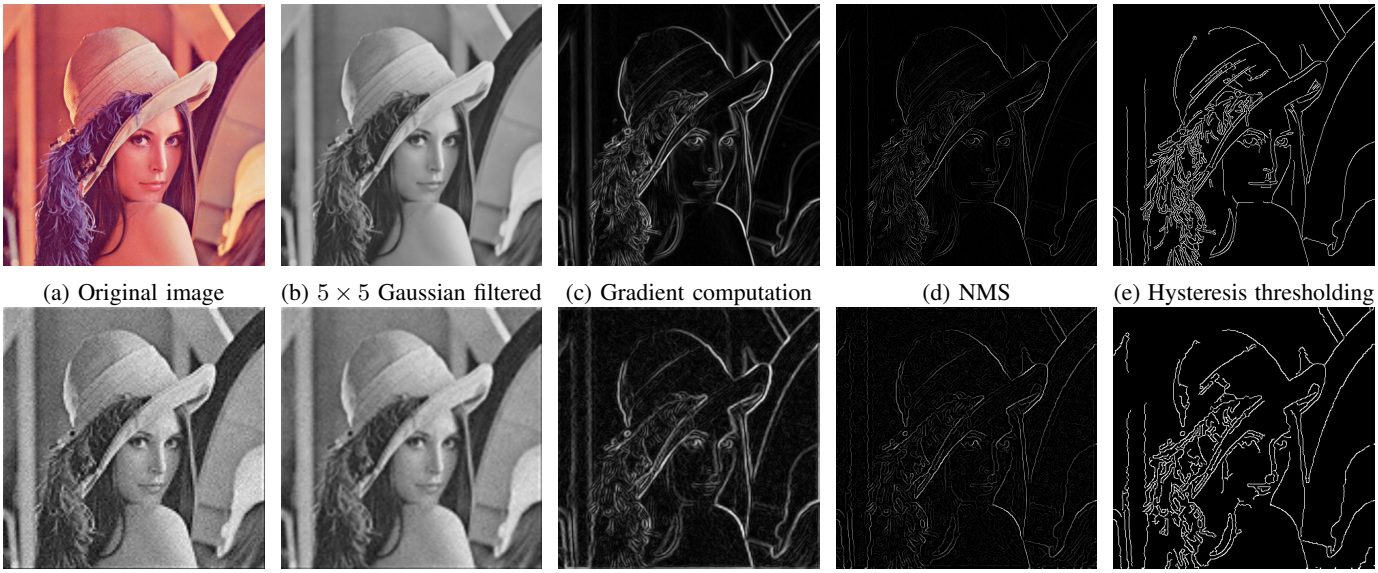
(c) 5×5 Gaussian filter

Fig. 12: Noise reduction
Literature suggests to use a 5×5 filter with $\sigma = 1.4$

I have tried a few combinations of operators (i.e, Prewitt, Sobel, etc...) and thresholding methods (i.e., Basic one, Otsu's, etc...) and I have extracted information on how each of those responded to the problem. I tuned the threshold boundaries to find which were the optimal ones for the implementation and in the end I settled for a value in the range $70 - 90\%$ as the best high to low ratio. Before coming onto that, though, I want to present the full pipeline of Canny edge detector, and how the edges map is transformed at each operational stage, between a clean *lena* image and a noisy one, in Fig. 13. Indeed, since Canny algorithm was designed for ideal step edges it has a notable worse performance when in presence of noise, especially impulsive, since the Gaussian filtering doesn't help as much there.

Despite detecting some of the edges, Canny edge detector is clearly more accurate in the first set of images. False positives and negatives are an issue in a noisy setting. Some lines are indeed not straight as they should be, since the extraction of local maxima along the edge directions was disturbed by the noisy pixels which were miscounted as edge pixels. Many details across the hat are lost, since the Gaussian filter had to be a little more aggressive to smooth the image there, and some edge pixels are missed leaving most of the connected shapes open, while they shouldn't be.

Finally, I am going to sum up the last bit of the *Comparative analysis* paragraph from the Jupyter notebook into a table, which highlights the performance of my implementation of Canny edge detector against the library implementation from



(a) Original image (b) 5×5 Gaussian filtered (c) Gradient computation (d) NMS (e) Hysteresis thresholding

Fig. 13: Canny on *lena.png* clean (↑) and noisy (↓)

OpenCV, evaluated over two different metrics: (i) Speed. (ii) False positives/negatives %. Refer to the Jupyter notebook to have a look at the respective edge maps. Speed is computed over the Canny pipeline presented in the above block schema. False positives/negatives % are instead average percentages gathered after applying Canny edge detector on all the four sample images I used as test cases, by comparing the edge maps I obtained with the one from OpenCV's library, considered for the time being representative of the true edges.

Canny implementation		**Metrics**		
Thresholding method	*Gradient operator*	*Speed (s)*	*False positives %*	*False negatives %*
OpenCV library		0.004	-	-
Otsu	Sobel	0.870	0.098	14.228
Otsu	Prewitt	0.772	0.099	14.224
Basic thresholding	Sobel	0.810	0.106	14.148
Basic thresholding	Prewitt	0.796	0.106	14.155

Table I: Canny performance comparisons

As I expected, my implementation was much slower than OpenCV's one, but surprisingly the values among the different methods I tried are quite close. Nonetheless, Sobel operator makes the computation slightly slower than Prewitt's, but yielding slightly better accuracy. Despite that, both failed at gaining small details, and thus missed some edges here and there.

A. Bonus experiment

I also tried just for fun how to apply the Canny algorithm to extract the edges out of a real time video capture from OpenCV, but my implementation was too slow for the demanding frame rate and often crashed. Hence, I'll just leave here what I was able to do using OpenCV's Canny implementation that, as you can see from Table I, is much much faster.

(a) Me praying the day before DSIP exam

(b) Edges map

REFERENCES

[1] N.R. Pal, S.K. Pal, A review on image segmentation techniques, PR 26 (9) (1993)1277–1294.
[2] H.D. Cheng, X.H. Jiang, Y. Sun, J. Wang, Color image segmentation: advances and prospects, PR 34 (12) (2001) 2259–2281.
[3] D. Ziou, S. Tabbone, Edge detection techniques—an overview, Int. J. PatternRecognit. Image Anal. 8 (4) (1998) 537–559.
[4] X. Wang, Laplacian operator-based edge detectors, PAMI, IEEE Trans. 29 (5)(2007) 886-890.
[5] J.F. Canny, A computational approach to edge detection, IEEE T-PAMI 8 (6)(1986) 679-698.
[6] J.S.Huang, D.H.Tseng, Statistical theory of edge detection, CGIP43(1988) 337-346.
[7] D.R. Martin, C. Fowlkes, J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues, IEEE T-PAMI 26 (5) (2004) 530-549.
[8] M. Morrone, R. Owens, Feature detection from local energy, PRL 6 (1987) 303-313.
[9] J.L. Marroquin, J.E. Figueroa, M. Servin, Robust quadrature filters, J. Opt. Soc. Am.A 14 (4) (1997) 779-791.
[10] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, Proceedings of the Sixth International Conference on Computer Vision, Narosa Publishing House, Bombay, 1998, p. 839.
[11] I. Pitas, A.N. Venetsanopoulos, Order statistics in digital image processing, Proc. IEEE 80 (12) (December 1992) 1893-1921.
[12] R. Gonzalez, R. Woods, Digital image processing, 3rd Edition (2007)
[13] B. Wang, S. Fan, An improved Canny edge detection algorithm, IEEE (2009)
[14] I. Kamanga, An adaptive approach to improve Canny method for edge detection, IJSR (2017) 164-168.